Design choices for agent-based control of AGVs in the dough making process

Martijn Mes, Matthieu van der Heijden, Jos van Hillegersberg

March 7, 2007

Abstract

In this paper we consider a multi-agent system (MAS) for the logistics control of Automatic Guided Vehicles (AGVs) that are used in the dough making process at an industrial bakery. Here, logistics control refers to constructing robust schedules for all transportation jobs. The paper discusses how alternative MAS designs can be developed and compared using cost, frequency of messages between agents, and computation time for evaluating control rules as performance indicators. Qualitative design guidelines turn out to be insufficient to select the best agent architecture. Therefore, we also use simulation to support decision making, where we use real-life data from the bakery to evaluate several alternative designs. We find that architectures in which line agents initiate allocation of transportation jobs, and AGV agents schedule multiple jobs in advance, perform best. We conclude by discussing the benefits of our MAS systems design approach for real-life applications.

1. Introduction

In recent years there has been a growing interest in distributed intelligent manufacturing due to the necessity of greater adaptability and flexibility to changes in the market demand. Agent technology is considered an approach that holds high promises for developing such systems (Jennings 1998). MASs are believed to be particularly suited for decentralized systems in real-time and dynamic environments. Because problems are solved locally, these systems should (1) be able to deal with a high level of complexity (2) require less information exchange then central control methods (3) respond fast to unexpected events and (4) reduce system nervousness compared to global optimization (which may lead to completely modified plans in case of minor information updates). In an earlier paper (Mes et al. 2007) we have described the benefits of decentralization and we have compared the performance of a basic multi-agent system with two central scheduling heuristics. Using a case study on an underground AGV system around Schiphol Airport Amsterdam, we found that a properly designed multi-agent system performs as good as or even better than central scheduling methods.

However, as multi-agent systems (MAS) are starting to find their way from laboratory settings to real-life manufacturing, full life-cycle methodologies are needed to support MAS development. Methodologies that have recently been introduced are built upon concepts from object-oriented software engineering and artificial intelligence. These methodologies generally provide guidelines for identifying agents, their roles, responsibilities and interaction protocols (Jiao et al. 2005). However, when applying these guidelines, several alternative designs for MAS can be derived. Designs may vary in the roles and responsibilities assigned to agents, the level of intelligence of the agents (forecasting and learning behaviour), and the interaction protocols selected. Current MAS methodologies lack a mechanism to evaluate such design-choices and provide only limited support to the designer in selecting the preferred design for implementation. Therefore, we propose to extend current MAS methodologies by multi-agent discrete event simulations. These simulations provide insight in the effect of MAS design choices on system quality aspects such as logistical performance (handling and delivery times), scalability, computing time, communication cost and robustness of the system.

We demonstrate and test this approach by applying it to a real life project; the design and development of MAS for manufacturing of biscuits at the industrial bakery Merba in the Netherlands. Merba produces a wide range of cookies for the Dutch and international market and is among Europe's largest producers of American chocolate chip cookies.

The goal of the project is the automation of the dough making process. Currently, employees collect ingredients for dough manually into barrels and move these barrels between the various processing locations. This manual process has a negative effect on the labour conditions, the product quality and the traceability of ingredients. Product quality problems arise from deviations in rising times of dough and amount of ingredients. Also, human body contact with the dough is inevitable. In order to overcome these problems and to achieve cost reductions, Merba aims at a fully automated dough production process using Automated Guided Vehicles (AGVs). To achieve a reliable and flexible AGV system, Merba aims at implementing MAS for scheduling transportation tasks.

During 2006, we have worked with Merba in implementing this system. In carrying out this project, we found that current MAS development methodologies aid in creating various alternative MAS designs, but do not provide sufficient support to select the preferred design for implementation. Therefore we applied multi-agent discrete event simulations in addition to the conceptual design stages proposed in common MAS design methodologies. In this paper we demonstrate and evaluate this approach. We investigate in what way design choices effect logistic and system performance. We thus follow a design science approach (Hevner et al. 2004) in which the artefact (the MAS methodology) is extended by adding simulation and evaluated in a field project.

The remainder of this paper is structured as follows. In the next section we give an overview of related literature. The Merba setting and the requirements of the project are described in Section 3. In section 4 we present our extensions to current methodologies to design MAS. We describe the resulting alternative agent-based designs in Section 5. We present our simulation experiments of the alternative designs in Section 6. Present some extensions (Section 7) end up with conclusions (Section 8).

2. Literature

Wooldridge and Jennings (1995) define an agent as a computer system that is situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives. An intelligent agent is further required to be proactive and social (Wooldridge 2002).

Agent technology has been used for a vast range of applications, ranging from e-mail assistants to air traffic controllers, see Jennings (1998). Recently, agent-based control architectures have been suggested as alternatives to traditional manufacturing control techniques (McDonnell et al. 1999). One of the earliest agent-based manufacturing systems, called “yet another manufacturing system” (YAMS), was developed by Parunak (1987). They consider a hierarchical production system in which each node (factory, manufacturing cell, workstation, and machine) is represented as an agent. Each agent has a collection of plans and negotiates with lower level agents to assign production tasks. In a later paper, (Parunak 2001) presented AARIA in which the manufacturing resources (e.g. people, machines, and parts) are encapsulated as autonomous agents that are using a mixture of different scheduling techniques. This approach to represent manufacturing resources by agents is common in agent-based manufacturing systems. Coordination between the agents is usually achieved through negotiation or an auction protocol. For example Lin and Solberg (1992) introduce part agents and resource agents that negotiate with each other to achieve individual objectives. Maturana et al. (1999) also use resource agents and introduce mediator agents for coordination between agents. McDonnell et al. (1999) use three classes of agents: part managers, resource managers and information managers. Coordination is achieved using an auction protocol to construct complete plans for part production. In (Maione and Naso 2001) part dispatching decisions are made by part agents and machine agents. The proposed approach is effective and reactive to severe disturbances and changes in the manufacturing environment. Shen and Norrie (2001) present an approach that combines mediation and bidding mechanisms for agent-based dynamic manufacturing scheduling. For more references

on agent-based systems in the manufacturing area we refer to (Jennings 1998; Parunak 1998; Shen and Norrie 1999).

Most papers on agent-based control of AGV systems focus on routing (Wallace 2001; Frazzoli et al. 2005) and the MAS architecture (Heragu et al. 2002; Kim et al. 2002). Only a few papers have appeared on planning and scheduling decisions in agent-based AGV control. An early application can be found in (McElroy et al. 1989) where intelligent AGV agents bid for transportation of loads. Also closely related is the decentralized architecture of Liu (2002) for the coordinated control of AGVs. They compare their approach with a centralized approach and conclude that their system provides higher utilization and is more robust to fluctuations in processing times. Boucke et al. (2004) propose a negotiation protocol for flexible and decentralized allocation of transportation tasks. This approach consists of a continuous negotiation protocol where the allocation of tasks is continuously reconsidered until the task is actually started. Lau et al. (2003) describe AGV control for material handling in an automated warehouse. They present a self-organizing distributed system where schedules for transportation tasks arise from interactions between the AGVs.

We observe for both manufacturing and transportation systems that (i) the majority of research on agent-based planning and control focuses on the development of generic architectures/frameworks, (ii) usually only a single architecture is given without any quantitative selection from alternative designs (iii) case oriented research is often limited to a conceptual description, in combination with simulation based on artificial data rather than real-life data. The contributions of our paper are (i) to show that qualitative arguments and modelling guidelines in current MAS methodologies are insufficient to select a single “best” architecture for MAS (ii) to show how simulation can be used to help in this selection process (iii) to evaluate this approach by applying it in a real world setting.

3. Requirements for the Agent System

In order to come up with an AGV control system we first established the system requirements by performing interviews with the management of the bakery. The main requirement is a flexible production system that (i) can easily be adjusted to new product introductions or modifications in the bakery layout, and (ii) can react in real-time on process uncertainties like equipment failures, product quality problems and the arrival of rush jobs. For example, if the quality of a (part of a) batch is insufficient, additional dough has to be prepared, leading to insertion of new jobs in the dough preparation schedule. Because of the first issue, the management decided to use AGVs rather than pipelines for the transportation of dough

ingredients (as is common in industrial bakeries). Because of the second issue, the management prefers a multi-agent system for the logistic control of the AGVs.

From these requirements, using a MAS methodology, we designed several alternatives of a multi-agent system that we discussed together with our modelling assumptions with the management of the bakery. To evaluate the design alternatives, we implemented a prototype in a simulation environment and collected the input data for this experiment at the bakery. We provided these alternatives together with the simulation results to the bakery management in order to make the final decision.

In the next subsections, we describe the dough preparation process at the industrial bakery, our model of the physical process with our basic assumptions, and the decisions involved in planning and control.

3.1. Process description

The production process at this bakery consists of three phases (1) preparation of dough (2) baking of cookies and (3) packing and storage. Our focus is on the first phase, the preparation of dough. Dough is produced in barrels that are essentially the same. AGVs transport the barrels between the various locations in the dough preparation process, see Figure 1.

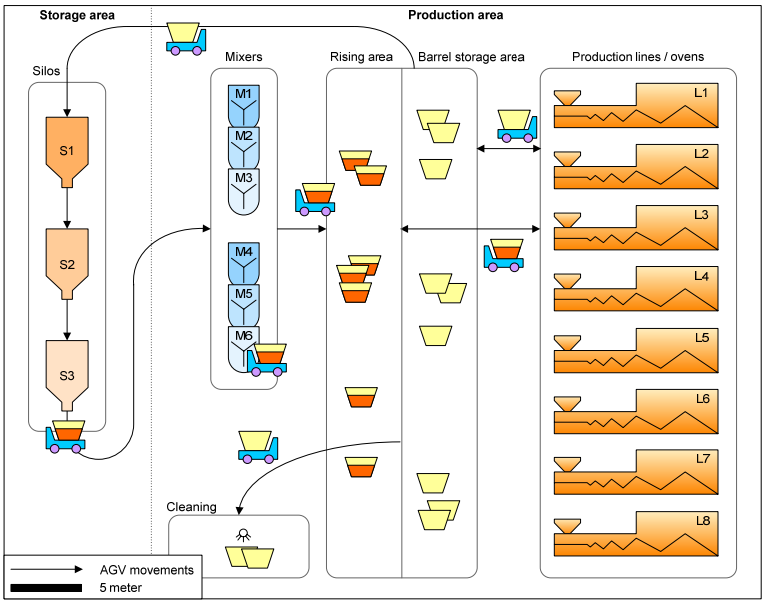

Figure 1 - Factory layout

The process always starts with a dough production request generated by the Manufacturing Execution System (MES). Each dough request is restricted by an earliest- and latest delivery

time of the dough at the line. The timing of dough requests depends on the day planning and the packing department where cookies undergo a quality check.

First, we have to find a suitable barrel for the dough request. This can be a barrel that has been used before for a similar dough type or a barrel that has been used for an incompatible dough type that has to be cleaned first at a special cleaning area. An AGV picks up the barrel and moves it towards a storage area consisting of three silos. Each silo may contain multiple ingredients and the silos have to be visited in a fixed order (displayed by S1 – S2 – S3 in Figure 1). The AGV positions its barrel below these silos to collect the ingredients. The time spent at each silo is the same for all dough types. Next, the AGV moves the barrel to a mixer. A single unique mixer is assigned to each dough type, but a mixer may process multiple dough types. During mixing, also new ingredients might be added to the dough such as decoration or ingredients that may only be added just before finishing mixing (like chocolate chips). After mixing the ingredients, the dough has to rise before it can be put into the oven. Once the rising time has passed, an AGV moves the barrel to the production line. Each dough type is assigned to a single production line, but each production line may process multiple dough types. There are no setup times for switching between dough types. At the production line, the barrel is emptied in a feeder. This feeder slowly delivers the dough to a conveyer belt moving through the oven. Because a feeder can store more than one barrel of dough, dough can be delivered some time before it is actually needed, that is, before the earliest delivery time. The empty barrel will stay on the AGV or will be dropped at the barrel storage area.

The rising time plays an important role in planning and control of the dough preparation process. Rising starts when ingredients are mixed and stops when the dough is used at the production line. That is, rising continues if the dough is being transported by an AGV and while it is waiting in the feeder. However, each dough type has a minimum rising time at the rising area. Therefore, an AGV always waits at the rising area until the minimum rising time has passed, even if this leads to a violation of the latest delivery time. For each dough type, a best rising time is specified. The product quality depends on the deviation between the actual rising time and the best rising time. A key problem for the planning and control system is to balance product quality (deviation from the best rising time) and tardiness (w.r.t the latest delivery time).

3.2. Model

Planning of the dough preparation process consists of scheduling all transportation jobs. To describe this in more detail, we discuss in this section (1) generating dough requests (2)

translating dough requests to transport jobs (3) the overall goal function of the planning problem and (4) the model assumptions.

Generating dough requests

As mentioned in Section 3.1, dough requests are generated by the Manufacturing Execution System. We assume that dough requests arrive one by one according to some stochastic process (e.g. a Poisson process). Each dough request has the size of a single barrel and is characterized by a certain dough type and time-window restrictions. Each dough type has a unique mixer, production line, minimum- and best rising time. The time-window of a dough request consist of an earliest- and latest delivery time. The earliest delivery time is the time a line expects it can start processing the dough. If dough is delivered before this time, it has to wait in the feeder. Delivery after the latest delivery time is penalized.

Translating dough requests to transport jobs

From the process description in Section 3.1, we can distinguish five job types for the AGVs: (i) silos - mixer (ii) mixer - rising area (iii) rising area - production line (iv) production line – barrel storage area (v) barrel storage area – silos. At each location, an AGV may drop the barrel in order to carry out other transportation jobs during processing times for e.g. collecting ingredients and rising. However, it is not always practical to drop the barrel, because (a) dropping and picking up the barrel takes time and may cause waiting time for an AGV after processing is finished (b) the AGV is needed during several processing steps for technical reasons (for example: an AGV is needed to move the barrel from one silo to the other when collecting ingredients, so that the barrel cannot be dropped at the silos; then it is practical if the AGV can move the barrel to the mixer immediately after collecting the ingredients. For these reasons, the management decided that an AGV is not allowed to drop the barrel at the silos, the mixer and the production lines. As a consequence, we only have two job types for the AGVs: a *preparation job* (barrel storage area – silos – mixer – rising area) and a *delivery job* (rising area – production line – barrel storage area)

A preparation job may be scheduled immediately after release of a dough request, even if the corresponding line has already released other jobs that are not yet delivered. We decided to postpone releasing the delivery job until the corresponding dough has been delivered at the rising area, because (1) the earliest- and best delivery time of the delivery job is dependent on the uncertain delivery time of the dough at the rising area and (2) rising times provide enough flexibility to schedule the delivery job. Whenever an AGV delivers dough at the rising area, it informs the corresponding line about the actual delivery time so that it can release the delivery job.

Goal function

The overall goal is to minimize costs based on two costs drivers, deviation from the best rising time and tardiness with respect to the latest delivery time. We normalize the penalties for deviation in rising time to 1 per time unit. We use the relative costs α for one minute tardiness compared to one minute deviation from the optimal rising time. The management of the bakery can influence the planning by manipulating α as the relative importance of tardiness compared to deviation from the best rising time (timeliness versus product quality). Note that these penalty functions can easily be extended towards non-linear functions. Disadvantage of this is that it will be less intuitive to the managers.

Model assumptions

Throughout this paper we make the following assumptions:

1. Although all travel- and processing times may be stochastic; we assume that their means are known. Only the mean waiting times have to be estimated by the agents themselves.
2. We do not explicitly include traffic congestion in our model, but we can correct for traffic delays by adjusting the effective AGV speed.
3. All dough requests have to be handled, even if they are late.
4. An AGV can park at any location when it is idle.
5. We omit the processing times of dough at the production lines from our model. By using externally generated time windows we ignore possible interdependencies between subsequent dough deliveries at the same line with limited capacity.
6. We omit batteries from our model. We assume that recharging or swapping batteries takes place in the idle times of AGVs.
7. We omit cleaning of barrels for ease of presentation. We assume that whenever an AGV delivers dough at the production line, it drops the empty barrel at the barrel storage area. For each new dough request, there is a clean barrel available at the barrel storage area.

3.3. Planning and control

The goal of the bakery is to balance deviation from the best rising time and tardiness at the production lines. We operationalize this by minimizing the deviation from the best rising times plus α times the tardiness. The main functionality the AGV system should perform is to assign all transport jobs (preparation and delivery jobs) to AGVs and to schedule these jobs. In order to do so we have to reckon with (1) the minimal and best rising time of dough and (2) limited capacity (and therefore waiting times) at the silos and mixers.

Because we decompose our system into multiple agents, the main goal of the bakery has to be achieved by individual agents with individual goals. Here we face two difficulties (1) we have

to deal with multiple criteria and (2) goals of individual agents may differ from the main goal (or might even be conflicting). To deal with multiple criteria we introduced the relative costs α. An example of divergence in goals is that minimizing the costs of one dough delivery may have a negative effect on the costs for the next dough delivery. An AGV with the goal to minimize the tardiness and deviation from best rising times might incorporate extra waiting for a preparation job such that the expected rising time equals the best rising time. Because scheduling jobs has an impact on the future availability of AGVs, it might be the case that future jobs are delivered late. Therefore we enable the individual agents to value their capacity.

In the next section we describe alternative agent architectures to support the allocation and scheduling decisions.

4. Alternative Designs for the Agent System

According to Luck, McBurney et al. (2003), a suitable methodology for analyzing, designing and building multi-agent systems is a key factor to introduce agent-orientation as an engineering approach to the industry. Three well known methodologies are Prometheus (Padgham and Winikoff 2004), Gaia (Wooldridge et al. 2000) and MaSE (Wood and DeLoach 2000). Roughly speaking each of these three methodologies consists of the following steps:

1. Decomposition of the system into multiple functionalities.
2. Allocation of functionalities to agents.
3. Establishing interaction protocols between the agents.
4. Designing the decision making capabilities of the agents.

The terminology may vary, but the approaches have many similarities. The first step is usually achieved by listing all system goals and grouping related goals. These related goals, together with related data, triggers and actions, form functionalities. The main task here for the system designer is to decide among alternative decompositions. In the second step it is decided how these functionalities are allocated to agents. In the third step, we face several design choices such as the sequence of steps in an interaction protocol. In the last phase we have to design protocols for internal processing of the individual agents. This involves the way they react on triggers and incoming messages. In our approach, we call the first three steps the *architectural* design phase and the last step the *detailed* design phase. The architectural design phase is generally supported by Agent Oriented Methodologies. For the

Detailed Design Phase, support is currently lacking, especially to quantify the quality of the design. Therefore we apply simulation as a design technique to support this phase.

Although our proposed method is independent of the specific agent design methodology used, we select the Prometheus methodology and the Prometheus Design Tool. It is a practical, rather complete and easy to understand methodology that especially provides support to our design choices in the architectural phase, which we describe below. The detailed design phase is described in Section 5.

4.1. Architectural design phase

4.1.1. Decomposition of functionalities (step 1)

The main goal of the bakery, balancing the deviation from best rising times and tardiness (Section 3.3), has to be decomposed into multiple functionalities, which can be assigned to different agents. A functionality describes a behaviour, consisting of decisions and actions, together with relevant triggers and data (Padgham and Winikoff 2004). Here we focus on the decisions and ignore physical actions (drive, pickup etc) which are obvious. First, we create a network of connected goals, see (Padgham and Winikoff 2004). The main design choice here is to group these goals. To select reasonable groupings we use the standard software engineering criteria of coupling and cohesion. Coupling is the level of interdependency between functionalities, while cohesion is the level of uniformity of the goals in a functionality. After evaluation of different groupings we end up with three functionalities: AGV selection, dough preparation management, and dough delivery management. AGV selection is concerned with the selection of AGVs waiting in a queue before a silo or mixer. The last two functionalities are concerned with the allocation and scheduling of respectively preparation jobs and delivery jobs.

4.1.2. Allocation of functionalities to agents (step 2)

Next we have to allocate the functionalities to agents, which represent physical objects in the bakery. We have the following objects: AGVs, lines, silos and mixers. Besides an AGV agent and a line agent, we use a storage agent that combines the silos and mixers because an AGV will always visit a mixer directly after visiting the silos.

In order to assign functionalities to these agents we look at data and triggers. Functionalities may be triggered by actions of physical objects within the factory, which then form candidates for these functionalities. We also evaluate the data used and produced by different

functionalities. This of course requires an iterative approach, because at this point we can only guess where information is located and which information is necessary for decision making. Functionalities that share the same data source form candidates for allocation to the same agent because it requires less information exchange.

The AGV selection functionality has as triggers the arrival of an AGV at a silo or mixer, and finishing loading ingredients of mixing. Therefore we allocate this functionality to the storage agent. For the dough preparation and dough delivery functionalities, we decide to investigate allocation to either the line agent or AGV agent. If we allocate these functionalities to the line agent it will search for an AGV based on its triggers (e.g. it receives a dough request, or dough has been delivered at the rising area). If we allocate these functionalities to the AGV agent, then it will search for a job at all lines based on its triggers (e.g. it becomes idle). We choose to evaluate both allocations using simulation. In the remainder of the paper we refer to the situation where these functionalities are assigned to the line agent by line centric (LC) and the case they are allocated to the AGV agent by AGV centric (AC).

4.1.3. Interaction between agents (step 3)

Having allocated functionalities to agents, we now have to specify how agents exchange information in order to perform their given functionalities. Again we use Prometheus by building scenario's, interaction diagrams and protocols, see (Padgham and Winikoff 2004). Main difficulty is to establish suitable interaction sequences that describe (1) which agents communicate with which other agents and (2) the timing of communication. Given the different agent- and message types, we might end up with a large number of possible interaction sequences. Therefore we propose a stepwise approach. First we focus on the order in which agents are involved in an interaction sequence, which we indicate by a communication sequence. From this we derive communication schemes that describe who communicates with whom. Next we make a selection of suitable communication schemes. Finally we specify communication by describing the communication protocols. This results in several agent architectures which we evaluate using simulation.

We illustrate this approach only for the dough preparation management functionality. The interaction sequences for the other two functionalities are obvious because they require only two agent types.

Communication sequences

The initiator of a communication sequence is given by the agent that is responsible for the functionality under consideration. Given the two allocations of the previous section (LC and AC) and the three agents (AGV, line, storage) involved in the dough preparation management

functionality, we have 4 possible sequences. However we also have an option to discard some agents in the decisions processes. We decided also to consider communication sequences without the storage agent, which result in 2 additional sequences (1 for each allocation).

Communication schemes

An overview of all possible communication schemes for the dough preparation functionality is given in Figure 2. The initiator (agent at the first row) always communicates with the second agent in the sequence. The third agent however can be contacted either by the first or by the second agent. Each of these schemes provides a rough sketch of a possible protocol. Consider for example the 6th scheme: based on its triggers, the line agent is triggered by a dough request and generates a preparation job. The line agent contacts the AGV agents for offers to process this job. To generate an offer, each AGV has to decide when to the job should be started. Therefore, they communicate with the storage agent about available capacity at the silos and at the mixer.

Selection of communication schemes

In principle, each scheme from Figure 2 could be implemented. However, we select a few schemes for our numerical experiments using qualitative arguments. We may consider (1) the scarceness of resources in the communication sequences and (2) required information exchange in the communication schemes. The scarceness of resources is not unambiguous here because in a dynamic environment, such as the bakery, it may occur that at different moments, different resources will be the bottleneck. Therefore we focus on the expected information exchange. As a guideline regarding the information exchange we avoid schemes for which we expect to use the same information to make decisions compared to another scheme but require more information exchange.

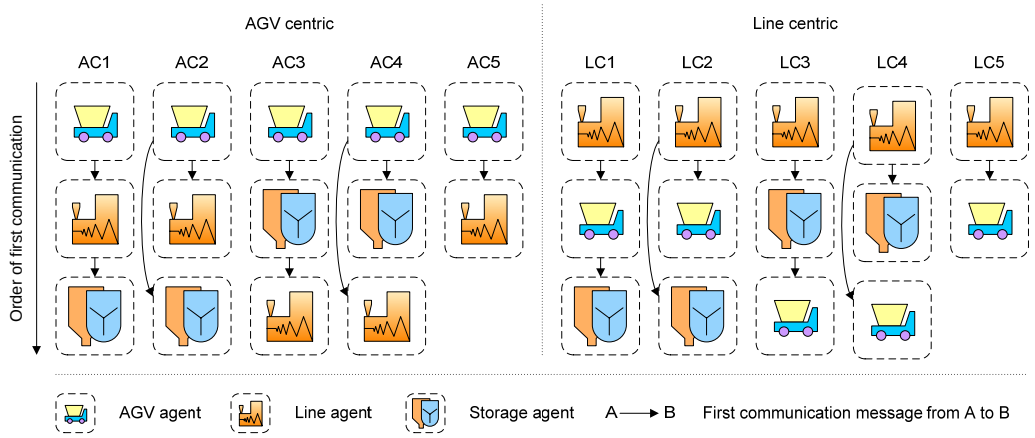

Figure 2 – Communication schemes

Basically, the dough preparation management functionality should support the following decision: which AGV should do which preparation job at what time. The three agents involved in this decision process have the following information that might support this decision. The line agent has knowledge about dough requests and dough deliveries. The AGV agent has knowledge about its availability and expected waiting times. The storage agent has knowledge about AGV arrivals. For both allocations (LC and AC) we select one communication scheme with three agent types using the following 2 observations.

1. If the first agent in a scheme communicates with the other two agents, it is required that the second agent provide all necessary information to the first agent. Otherwise, the first agent has to provide all necessary information to the second agent. In the AGV centric schemes, an AGV agent only inform the other players about its idle status. So, schemes AC2 and AC4 require more information exchange than respectively schemes AC1 and AC3. In the line centric schemes, AGVs may have schedules with multiple jobs, and the storage agent may have a schedule with multiple AGV arrivals. Because the line agents inform the other players only about a single job, schemes LC2 and LC4 require more information exchange that respectively schemes LC1 and LC3.
2. Communication with the line agent in the AGV centric schemes always involves communication with all line agents (because we want to find the most suitable job for a specific AGV), and communication with the AGV agent in the line centric schemes always involves communication with all AGV agents (because we want to find the most suitable AGV for a specific job). Therefore it requires less information exchange if the storage agent is used as second agent instead of third. So we skip schemes AC1 and LC1.

After applying these guidelines we end up with schemes AC3, AC5, LC3 and LC5. Note that we only skipped schemes for which there is an alternative that requires less communication in order to make decisions based on exactly the same information. Therefore, this choice does not affect the logistics performance. However, it has an impact on the responsibilities and decision making capabilities of the agents. In schemes AC3 and LC3, the storage agent plays a more central role compared to the skipped schemes. A possible disadvantage, we did not take into account, is that these schemes have a single point of failure.

Agent architectures

Next, we have to specify the communication protocols to be applied to the remaining communication schemes. The most common protocol between agents in both real applications and detailed simulations is the Contract-Net Protocol (CNP) (Parunak 1987). The CNP, introduced by Smith (Smith 1980), is a high level negotiation protocol for achieving efficient cooperation. This protocol consists of four steps: (1) an initiator sends a call for proposals to a

set of participants (2) the participants respond with a proposal (3) the initiator chooses the best proposal and awards a contract to the respective participant (4) the other participants are rejected. We use the CNP to support the dough preparation management and dough delivery management functionalities. For the AGV selection functionality we simply use a FCFS strategy, that is, the storage agent simply selects the AGV that arrived first in a queue. Given the remaining communication schemes we derive four architectures. Here, an architecture consists of a description of how agents react on triggers and exchange information with other agents.

In the agent centric architectures (AC3 & AC5), the job allocation process is triggered by an AGV that becomes idle. Then it might occur that a line receives a dough request while all AGVs are idle. In this case the dough will never be allocated to an AGV. Therefore, we also trigger an arbitrary idle AGV, if there is one, whenever a new job arrives.
In AC3 (Figure 3), the AGV informs the storage agent about its position whenever it becomes idle. In return, the storage agent sends a request to all lines to submit their job characteristics. After receiving the job characteristics, the storage agent selects the most suitable job, informs the corresponding line agent about the expected delivery time of this job, and informs the AGV agent where to move to and when.

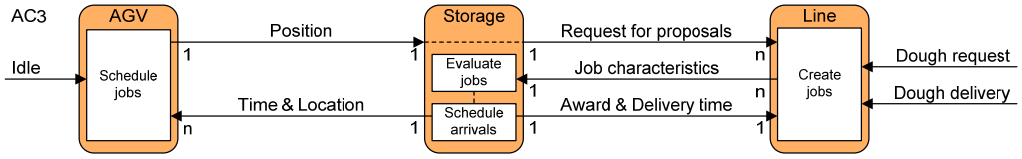

Figure 3 - AGV centric architecture AC3

In AC5 (Figure 4), the AGV sends a request to all lines to submit their job characteristics. After receiving the job characteristics, the AGV agent selects the most suitable job and informs the corresponding line agent about the expected delivery time of this job.

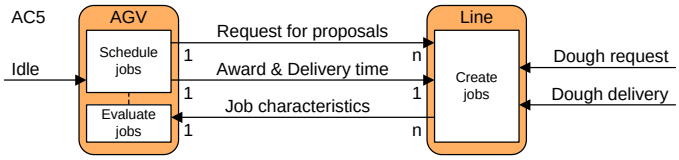

Figure 4 - AGV centric architecture AC5

In the line centric architectures (LC3 & LC5), preparation jobs are triggered by a line agent who receives a dough request. Delivery jobs are triggered when corresponding dough has

been delivered to the rising area. In LC3 (Figure 5), the line agent informs the storage agent about a new job. The storage agent sends a request to all AGVs. After receiving the requested information from all AGVs, the storage agent selects the most suitable AGV and informs both, the line agent and AGV agent.

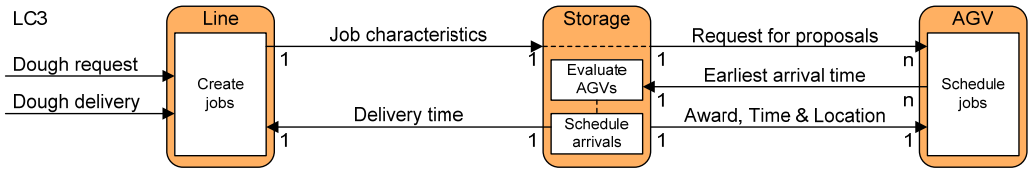

Figure 5 - Line centric architecture LC3

In LC5 (Figure 6), the line agent sends the job characteristics to all AGVs. Each AGV selects the best time to start this job and calculates a price. The line agent simply selects the AGV with lowest price and informs the winning AGV.

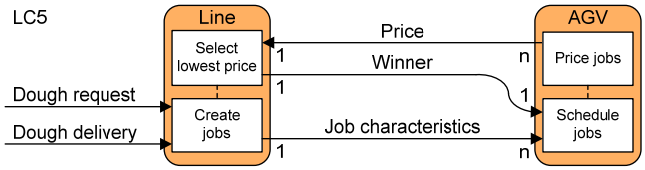

Figure 6 - Line centric architecture LC5

The decision making capabilities of the agents (illustrated by the white squares in the figures above) are described in Section 5.

4.2. Summary of the architectural design phase

In the architectural design phase we go from a main system goal through a sequence of the following steps: (1) decomposition into functionalities (2) allocation of functionalities to agents (3) establishing interaction protocols between the agents. The last step is achieved by (i) determining communication sequences for all allocations, (ii) determining communication schemes for each sequence and (iii) determining interaction protocols for all communication lines in each scheme.

This design approach enables us to avoid overlooking promising architectures. However, when there are a lot of resources (with corresponding agents), the number of possible schemes may become very large. In our case we have 12 schemes for the dough preparation management functionality and 2 schemes for the dough delivery management functionality.

Suppose we also incorporate the cleaning process of barrels. Then we have an additional functionality called barrel management and a cleaning agent. The total number of possible schemes is then 168. By going through a step-wise approach we are able to limit the number of possibilities in a structured manner. In our case we end up with 4 architectures.

5. Detailed design (step 4)

In this section we describe the following decision making capabilities that are required in the four alternative architectures: create jobs (5.1), evaluate jobs (5.3), evaluate AGVs (5.4), schedule arrivals (5.5), schedule jobs (5.6), and price jobs (5.7). In addition, we introduce a waiting strategy (5.2) that is required for scheduling and job evaluation, and end with a section on how agents estimate parameters (5.8).

5.1. Create jobs

As mentioned before, a dough request leads to a preparation job and a delivery job for the AGVs. The characteristics of these two jobs are determined from the characteristics of the dough request. We characterize a dough requests i by a dough type t_i, a release time a_i, a production line p_i, and an earliest- and latest delivery time at the production line denoted respectively by e_i and l_i. The dough type uniquely describes the following characteristics of a dough request i: the processing time t_i at the production line, a minimum rising time $r_{min,i}$ at the rising area, a best rising time $r_{best,i}$, and a mixer m_i. For notational convenience we subtract the travel time from the mixer to the rising area from the rising time. The rising time therefore starts upon delivery at the rising area and ends at the time the line start working on this dough.

When a line receives a dough requests at time a_i, it creates a preparation job with the following characteristics: a production line p_i, a mixer m_i and a best- and latest delivery time at the rising area, respectively denoted by b^p_i, l^p_i. For the best delivery time we use $b^p_i=max(a_i,e_i-r_{best,i})$ because we want to delivery this order as early as possible such that there is more flexibility for the corresponding delivery job. For the latest delivery time we use $l^p_i=l_i-r_{best,i}$ because delivery after this time will certainly result in penalties for the corresponding delivery job.

When a line receives a message, at time d, that dough has been delivered at the rising area, it creates a delivery job for this dough. This delivery job i is characterized by a production line p_i and an earliest-, best-, and latest delivery time of the dough at the production line, respectively denoted by e^d_i, b^d_i, l_i. The earliest delivery time is given by $e^d_i=d_i+r_{min,i}+\tau_i$, where

τ_i is the travel time between the rising area and the production line of job i. The best delivery time is given by $b^d_i = min(d_i + r_{best,i}, l_i)$.

5.2. Waiting strategy

To describe the waiting strategy we introduce a best starting time of a job. The best starting time of a preparation job provides the best time to start loading the ingredients at the first silo. The best starting time of a delivery job provides the best time to pickup the dough at the rising area.

In order to derive the best starting time, agents have to be able to make a trade-off between loss of capacity (waiting) and the direct costs caused by deviation from the best rising time or tardiness w.r.t. the latest delivery time. Therefore we introduce a cost factor β for the value of AGV capacity per unit time (see Section 5.8 for estimation of this parameter).

If ($\beta \geq 1$), then the costs for waiting are higher then the expected costs for deviation in rising times. The best starting time of a preparation job is given by the earliest arrival time of the AGV at the first silo. The best starting time of a delivery job is given by the maximum of the earliest pickup time (delivery time plus minimum rising time) and earliest arrival time of the AGV at the rising area.

Otherwise, if ($\beta < 1$), then it is better to wait if this results in less deviation from the best delivery time. The best starting time of a preparation job is given by the maximum of the earliest arrival time of the AGV at the first silo, and the best delivery time b^p_i minus the expected time between loading the ingredients at the first silo and the time to drop the barrel at the rising area. The best starting time of a delivery job is given by the maximum of the earliest arrival time of the AGV at the rising area, and the best delivery time b^d_i minus the travel time from the rising area to the production line.

5.3. Evaluate jobs

Job evaluation is used in the agent centric architectures to determine the job which should be handled first. Therefore we determine a priority value of each job. This value should reflect the priority of a job and the waiting time of an AGV doing this job. The total handling time is not a valid selection criterion because delivery jobs have shorter handling times than preparation jobs but may be equally important.

As input we need the job characteristics of a job i, and the expected earliest delivery times z_j for an AGV j to deliver these jobs. In AC3, jobs are evaluated by the storage agent. In order

to calculate the earliest delivery time of an AGV, the storage agent receives the current location of the AGV and the costs β. In AC5, jobs are evaluated by an AGV agent. In order to calculate the earliest delivery time, the AGV agent estimates the waiting times before loading- and mixing ingredients.

The priority value v_{ij} for an AGV j doing a job i, is a measure of the distances between the best- and latest delivery times, and the earliest possible expected delivery time z_j. If the earliest delivery time z_j is later than the best delivery time b_i^p, we add the difference to the priority value because other AGVs will do the job with even more penalties. If the earliest delivery time z_j is earlier than the best delivery time, then we subtract the difference because another AGV may do this job better at a later moment. The same holds for the difference between the earliest delivery time and the latest delivery time. If waiting is required due to minimum- or best rising time constraints, we subtract the value of waiting given by β times the waiting time. We have the following:

$$v_{ij} = \begin{cases} \left(z_j - b_i^p\right) + \alpha\left(z_j - l_i^p\right) - \beta \max\left(0, b_i^p - z_j\right) & \textit{preparation job} \\ \left(z_j - b_i^d\right) + \alpha\left(z_j - l_i\right) - \beta \max\left(0, b_i^d - z_j\right) & \textit{delivery job} \end{cases} \quad \textbf{(1)}$$

After calculation of all priority values, the job with highest priority will be selected.

5.4. Evaluate AGVs

AGV evaluation is used in LC3 to determine the AGV that should handle a specific job. We use the same approach as for job evaluation. As input we need the expected delivery times of all AGVs and the job characteristics of the job. To calculate the earliest delivery time of a delivery job, the storage agent receives the earliest arrival time at the rising area from all AGV agents. To calculate the earliest delivery time of a preparation job, the storage agent receives the earliest arrival time at the first mixer from all AGV agents. The AGV j with highest priority value v_{ij} for a specific job i, calculated with Equation 1, is selected.

5.5. Schedule arrivals

In AC3 & LC3, the storage agent maintains a schedule of AGV arrivals. This schedule consists of the following AGV handling records:

> *[AGV, Earliest arrival time, Scheduled starting time, Mixer, Arrival at mixer, Waiting time at mixer, Departure time, Best departure time, Latest departure time]*

Initially these times are random variables. For simplicity of the planning, we decide to use the expectations only. These expected times are updated at three events (1) whenever an AGV starts loading ingredients at the silos, (2) when an AGV leaves the mixer and (3) when a new AGV arrival is scheduled. When an AGV leaves the mixer the corresponding record will be deleted. Whenever the storage agent or AGV agent decides about a scheduled starting time, the storage agent adds a record to his schedule and updates the times of all records. When scheduled starting times of other AGVs are changed, they are only communicated to AGV agents at the moment they schedule a new job (earlier is not necessary).

The AGV arrival schedule has two purposes, earliest arrival scheduling and best arrival scheduling. Earliest arrival scheduling is used by the storage agent in AC3 to calculate the earliest delivery time of a preparation job. Best arrival scheduling is used in architectures AC3 & LC3 to schedule the starting times of AGVs such that the expected penalties are minimized. For both purposes, the storage agent needs to know the earliest arrival time of the AGV, the costs β, and the best-, and latest departure times from the mixers. The only distinction between the two purposes is that in case of earliest arrival scheduling, we set the best departure time equal to the earliest departure time (based on the earliest arrival time and zero waiting times at the silos and mixers).

The storage agent will schedule a new AGV arrival as close as possible to the best departure time, moving some jobs earlier (without violating the earliest arrival time restrictions) and moving other jobs forward. Therefore the storage agent evaluates the following situations: (1) for each insertion position after the first job the new arrival is scheduled directly after the previous arrival (possibly moving further orders forward) and (2) for each insertion position before the last arrival, the new arrival is scheduled as close as possible before the next job (moving earlier orders backwards and possibly moving further orders forward if preceding arrivals cannot be moved further backwards and (3) the delivery time of the new job is scheduled as close as possible to the best delivery time while moving the other arrivals.
If the current AGV arrival schedule contains n arrivals, then we have $2n+1$ possible insertion positions of the new AGV arrival. The alternative schedule with lowest costs will become the temporal schedule belonging to a new AGV arrival with certain preparation job.
In case of earliest arrival scheduling, the temporal schedule is only used to provide the earliest departure times for different jobs. In case of best arrival scheduling, the storage agent replaces the current schedule with the temporal schedule derived for the most suitable job or AGV.

5.6. Schedule jobs

In AC3, an AGV has only one job at a time. The storage agent determines the best starting time for this job (Section 5.2), calculates the loading- and mixing times in case of a preparation job, and informs the AGV where to be at what time.

Also in AC5, an AGV has one job at a time. This time the AGV determines the best starting time for a job (Section 5.2). Because, loading- and mixing times are not communicated with the storage agent, the AGV agents estimate the expected waiting times at the first silo and the mixer. The expected waiting time is subtracted from the best starting time.

In LC3, an AGV schedule may contain multiple jobs. Each time a new job arrives, the AGV add this job to the end of its schedule and determines the best starting time for this job (Section 5.2). The waiting times at the silos and mixers are calculated by the storage agent who maintains a schedule of AGV arrivals.

Also in LC5, an AGV schedule may contain multiple jobs. Again, AGVs may use a scheduling method, denoted by *append* scheduling, where new orders are always added to the end of the schedule. However, this time it has more freedom to schedule its own jobs because they are not communicated with the storage agent. Therefore we also consider an *insertion* scheduling method where a new job can be inserted at any position in the current schedule without altering the order of other jobs (like we did in the previous section with the AGV arrival schedule). For a given order of jobs, AGVs calculate the best starting times (Section 5.2). Because loading- and mixing times are not communicated with the storage agent, AGVs estimate the waiting times at the first silo and the mixer and subtract this time from the best starting time. AGVs may update their schedule at the following moments: arrival at some destination (line, silos, mixer, rising area), finishing an action (pickup barrel, drop barrel, loading, mixing), during bid calculation and after receiving a grant.

5.7. Price jobs

In LC5, AGVs have to price jobs and provide this price to the line agent. The price is given by the marginal costs of appending or inserting a new job in its current schedule. Depending on the scheduling method, the AGV agent can schedule the new job at different positions in the current schedule. We will indicate the current schedule by Ψ^* and we write Ψ^n for schedule alternative n, where the new job φ is inserted after job n ($1 \leq n \leq |\Psi^*|$). For each insertion position we also have to schedule the optimal starting times of all jobs. For example, suppose the new job is added directly after delivery of the last job in the current schedule and the new job is delivered after its due time, then we might remove unnecessary waiting times

for the previous jobs. Therefore we solve a simple linear program for each alternative schedule (see Appendix).

The total cost of a schedule Ψ is denoted by $V(\Psi)$. The bid price of a vehicle is given by the difference in costs between the cheapest alternative schedule and the current schedule:

$$P = \min_{n} V\left(\Psi^{n}\right) - V\left(\Psi^{*}\right) \quad \textbf{(2)}$$

The value of a schedule is given by the sum of the deviations in best delivery times, α times the total lateness and β times all waiting-, travel- and handling times (a formal expression can be found in the Appendix).

5.8. Parameter estimation

In order to perform their tasks, AGV agents have to estimate some parameters. In all architectures they estimate the costs β per unit time. In AC5 and LC5, they also estimate the waiting times before loading and mixing ingredients.

In order to estimate the variables, we use an exponential smoothing procedure (Silver 1998) where a learning rate γ is introduced as a weighting factor that determines the extent to which the current observation is to influence an expected value of an internal parameter. The meaning of the learning rate in this procedure is that when γ is close to one, the new forecast will be based almost exclusively on the last observation. Conversely, when γ is close to zero, the new forecast will be similar to the previous one.

The value of β is calculated by the AGV agent based on the average penalties paid per time unit. The logic behind this is that if we wait an extra time unit, this AGV will be available one time unit less, which possibly results in one time-unit of extra penalties. We use the exponential smoothing procedure to incorporate fluctuations in the average penalties. To avoid unstable behaviour we smooth the penalties per period instead of penalties per job. An extension for learning β is discussed in Section 7. Waiting times are updated after each visit at the first silo or mixer.

6. Simulation

The goal of this simulation study is twofold: (1) to find out which agent architecture can be used best at the bakery and (2) to demonstrate the impact of design choices on the system performance under different parameter settings. In this section we subsequently describe our fixed simulation settings, the experimental factors and the results.

6.1. Simulation settings

The bakery produces over 100 dough types. For ease of presentation, we aggregated these dough types into one fictive dough type per production line based on historical data of all dough requests. These virtual dough characteristics are given in Table 1. All times are given in minutes.

Line	TBO	Look-ahead l_i-a_i	Min rising time	Best rising time	Mixer
L1	30	50	15	20	M1
L2	60	50	15	20	M1
L3	15	70	20	30	M2
L4	30	70	30	52	M3
L5	30	70	30	52	M4
L6	30	70	30	52	M5
L7	15	70	30	52	M5
L8	30	70	30	45	M6

Table 1 - Dough request characteristics

Production runs 5 days per week, 24 hours per day. Every week, production starts Monday morning at 4:00 hour. The last batch is released to the dough preparation process on Saturday morning at 4:00. Because the system starts and ends empty each week, we have a terminating simulation. We apply the consider one week as a replication for our simulation experiments. We assume that the release of dough requests follows a Poisson process with mean number of jobs per hour per production line as given in Table 1. These figures have been derived from historical data on peak days of the bakery.

All AGVs have a constant speed of 1 m/s. For simplicity, we assume that AGVs can always travel in a straight line (shortest distance) from one object to another. We add half a minute to all movements to incorporate the time it takes for an AGV to turn. The time to pickup or drop a barrel is 30 seconds and the loading time for ingredients is 2 minutes per silo. The time for mixing is 11.9 minutes at mixer 1, 11.6 minutes at mixer 2 and 5.3 minutes at the other mixers. The distances between all objects can be calculated from Figure 1. For all experiments we use 7 AGVs, a penalty factor $\alpha=10$, and a smoothing factor $\gamma=0.05$. The number of AGVs is chosen such that all dough requests can be handled (not necessarily in time). In our simulation experiments we have seen that the choice for the penalty- and smoothing factor does not affect the relative performance of the alternative architectures. As overall performance measures we use (i) the penalty costs for job tardiness and deviation between actual and best rising time, (ii) number of communication messages, and (iii) the computation time. The number of communication messages provides an indication of the

network load. The computation time is measured per job assignment, taking into account parallel computation. We implemented the agent architectures in the object oriented simulation package eM-Plant and performed experiments on a Pentium IV processor 3.4GHz. All performance measures are calculated as weekly averages.

We choose the number of replications (weeks) needed in our simulation experiments such, that a 95% confidence interval for the total costs per work week shows a relative error of at most 5%. We found that 10 replications are sufficient for all scenarios.

6.2. Experimental factors

The experimental factors can be found in Table 2. We evaluate the 4 different agent architectures, and for architecture LC5 we consider the two scheduling methods. The stochasticity describes the uniform deviation around the mean handling- and travel times. So a deviation of 20 will result in handling- and travel times between 0.9 and 1.1 of the normal value. We include this factor to examine the impact of uncertainty because of possible congestion effects (which architecture and planning method is most robust?). Next, we consider three fractions that describe the deviation from the standard settings (Section 6.1). These factors will be examined one at a time.

Factor	Values
Architecture	AC3 / AC5 / LC3 / LC5
Scheduling in LC5	Append (LC5a) / Insert (LC5i)
Stochasticity (%)	0 / 8 / 16 / 24 / 32 / 40 / 48
Fraction TBO	0.90 / 0.95 / 1.00 / 1.05 / 1.10 / 1.15 / 1.20
Fraction handling times	1.00 / 1.08 / 1.16 / 1.24 / 1.32 / 1.40 / 1.48
Fraction look-ahead	0.8 / 0.9 / 1.0 / 1.1 / 1.2 / 1.3 / 1.4

Table 2 - Experimental factors

The fraction TBO provides the fraction of the mean time between subsequent order arrivals compared to the default values as given in Section 6.1. The fraction handling times describe the increase in handling time for silo 1 compared to the default value from Section 6.1. The handling times of the other two silos is decreased by half of this amount such that the total handling time at the storage department will be the same. In our case, a value of 1.4 means a handling time of 2 minutes and 48 seconds at silo 1 and 1 minute and 36 seconds at the other silos. We use this factor to investigate the effects of longer queues at the storage department. The fraction look-ahead is a multiplication factor for the look-ahead values from Table 1.

6.3. Results

In the first 4 experiments, we examine the performance of the different architectures in terms of penalties on tardiness and deviation from the best rising time.

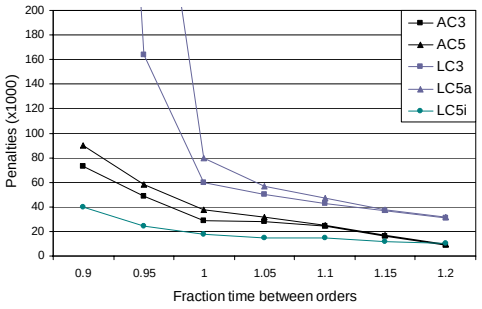

Figure 7 - Varying time between orders

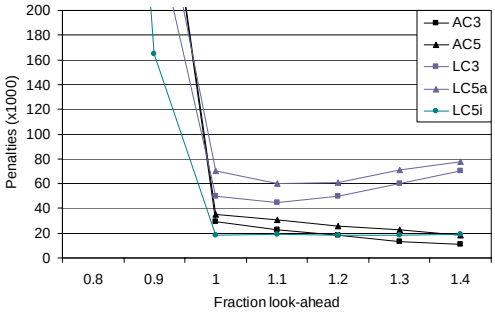

Figure 8 - Varying look-ahead

In the *first experiment* we vary the time between orders (Figure 7). We see that architectures LC3 and LC5a, where new jobs are added to the end of AGV schedules, are less robust against increasing number of orders. Architecture LC5i performs best in most situations. However with decreasing number of orders, the AGV centric architectures may become in favour. In the *second experiment*, we vary the look-ahead of jobs (Figure 8). We see a similar behaviour in which the AGV centric architectures are better with increasing look-ahead. The reason for this is that increasing look-ahead leads to longer schedules which may result in less flexibility. This is especially true in case of append scheduling, where also rush jobs have to be added at the end of the schedule. With decreasing look-ahead, the time becomes too short for AGVs to delivery the jobs on time.

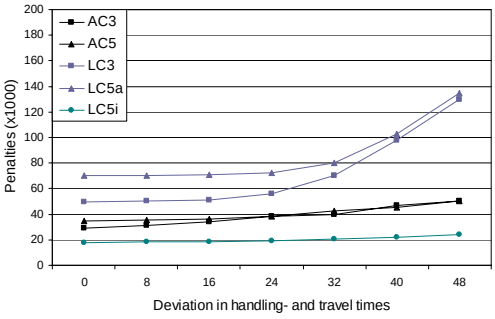

Figure 9 - Varying deviation

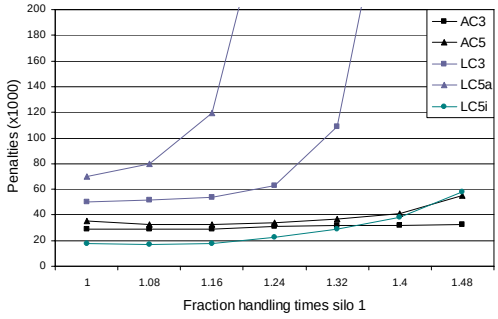

Figure 10 - Varying handling times

In a *third experiment*, we investigate the effect of uncertainty in the handling- and travel times (Figure 9). As expected, penalties increase with increasing uncertainly for all architectures.

We see that with increasing uncertainty, scheduling the AGV arrivals becomes less useful. In a *fourth experiment*, we investigate the effects of congestion at silo 1 (Figure 10). We see the performance of architecture AC3 remains the same with increasing congestion, while the costs of all other architectures increase. We also see that with increasing congestion, it becomes more useful the scheduling the loading- and mixing times (AC3 & LC3).

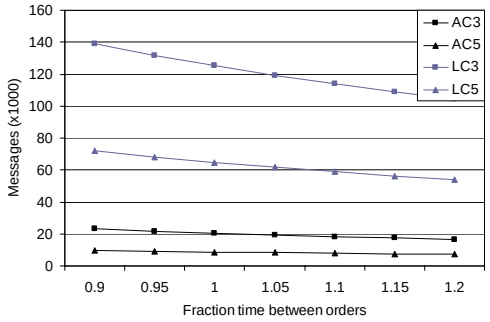

Figure 11 - Communication messages

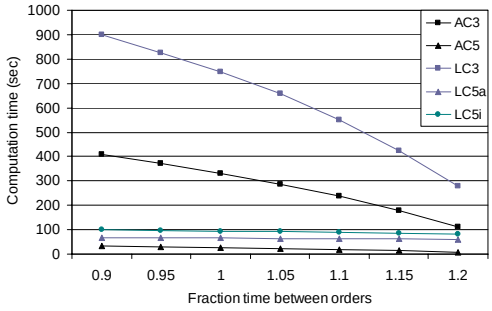

Figure 12 - Computation time

Next, we consider the number of communication messages and computation time as performance indicators. In the *fifth experiment*, we vary the time between orders and investigate the number of communication messages of the different architectures (Figure 11). Note that the number of communication messages is the same for both scheduling methods in architecture LC5. We see (1) the number of messages decreases with decreasing number of orders, (2) communication with the storage agent (LC3 & AC3) requires much more communication because loading- and mixing times have to be communicated for every order with every schedule update, and (3) the line centric architectures require the most amount of communication. In a *sixth and final experiment* we show the impact of a varying time between orders on the computation time (seconds) in Figure 12. The results are obvious: (1) scheduling loading- and mixing times, increases the computation time and (2) the computation time for all architectures decreases with increasing time between orders. Note that computation time is measured based on a parallel implementation. For architectures AC3 and AC5 this does not make a difference because we have sequential decision processes. However, in architectures LC5a and LC5i, most computation is done in parallel by all AGVs when they try to schedule a new job received by a line agent.

We conclude that architecture LC5i perform best in almost all situations, although it requires an intermediate amount of communication and computation time. However, the AGV centric architectures come in favour in case of (1) decreasing number of orders, or (2) increasing look-ahead or (3) increasing congestion at the silos.

The data used for the experiments described above is based on real factory data in which we changed one factor at a time. We did not use a full factorial design because this would be beyond the scope of this paper. However, we also examined the effect of several other experimental factors. We found that for each architecture, there exists at least one instance in which it performs best. For example, if the minimum rising times of dough are very small, the time-windows are very tight, and we have more congestion at the silos, the architecture LC3 performs best.

7. Some extensions

In Section 4.1.3, we mentioned the FCFS selection strategy at silos and mixers. An improvement can easily be made by using an auction protocol to select an AGV from the queue. Each AGV in a queue has to submit a bid consisting of the increase in penalties if it has to wait one turn. One turn here is the expected time until the next AGV may leave the queue. The AGV with highest bid will be handled first.

In Section 5.8, we calculated β as being the average penalties paid per unit time. An alternative is to calculate β as being the average revenue per time unit. In order to generate revenues we make use of a reverse Vickrey auction. In this auction the lowest bidder wins the auction and receives the price of the second lowest bidder. This auction type has received particular attention within the multi-agent community because it possesses a dominant strategy to bid one's true valuation (Vickrey 1961). This pricing strategy is especially suitable for architecture LC5 where AGV agents calculate a price and the AGV with lowest price is selected. The logic of calculating β, as being the average revenue per time unit, is that waiting an extra time-unit will decrease the period after this job with one time-unit. So, expected revenues decrease with the expected revenue of one time unit. But revenues also provide valuable information about future penalties. Let's consider an AGV that just won an auction. Its revenue for the new job resembles the increase in expected penalties if not he, but the second best bidder had won this auction. Because AGVs plan jobs in the future, this information provides insight in future network pressure. In other words, the last observation of revenues provides a better estimate of future penalties than the last observation of penalties already paid.

8. Conclusions

During our field project at Merba bakeries, we found that current MAS development methodologies do not provide sufficient support to select the preferred design for

implementation. Alternative designs vary in roles and responsibilities assigned to the agents, the level of intelligence of the agents, and the interaction protocols. To illustrate this, we considered a simplified part of the dough production process. By using a stepwise approach, built upon existing MAS development methodologies, we already derived eight alternative designs for this part only. By using qualitative arguments, we were able to reduce this to four alternative designs. In order to select the preferred design for implementation we used multi-agent discrete event simulation. This simulation provided us insight in the effect of our MAS design choices on system the performance in terms of delivery punctuality, product quality, robustness, amount of communication, and computation time of the different agents. That qualitative arguments are not sufficient is shown by our simulation study where we it appears that each alternative design have its own advantages. A practical way of dealing with these results is to use a combination of different control mechanisms. Based on the system status, AGVs might use another scheduling technique or the architecture itself may even be changed dynamically. Suppose for example that we are using architecture LC5i. Whenever we observe increasing congestion, we might temporarily switch to an AGV centric architecture. This adaptability of the system design is part of our future research. In addition, we want to investigate impact of MAS design choices on a broad class of performance indicators such as flexibility, scalability, adaptability and extensibility.

9. Acknowledgement

This research is supported by the BSIK project "Transition Sustainable Mobility" (TRANSUMO). The authors thank the management of Merba, J. den Hartog and W. Boerman, for their time and support.

10. References

Boucke, N., D. Weyns, T. Holvoet and K. Mertens (2004). Decentralized Allocation of Tasks with Delayed Commencement. Second European Workshop on Multiagent Systems, EUMAS, Barcelona, Spain.

Frazzoli, E., L. Pallottino, V. G. Scordio and A. Bicchi (2005). Decentralized Cooperative Conflict Resolution for Multiple Nonholonomic Vehicles. Proc. of the AIAA Conf. on Guidance, Navigation, and Control, San Francisco, CA.

Heragu, S. S., R. J. Graves, B. Kim and A. S. Onge (2002). "Intelligent Agent-based Framework for Manufacturing Systems Control." IEEE Transactions on Systems, Man, and Cybernetics **32**(5): 560-573.

Hevner, A. R., S. T. March, J. Park and S. Ram (2004). "Design Science in Information Systems Research." MIS Quarterly **28**(1): 75-105.

Jennings, M. J., K. Sycara, M. Wooldridge (1998). "A roadmap of agent research and development." Autonomous Agents and Multi-Agent Systems **1**(1): 7-38.

Jiao, W., J. Debenham and B. Henderson-Sellers (2005). "Organizational models and interaction patterns for use in the analysis and design of multi-agent systems." Web Intelligence and Agent Systems **3**(2): 67-83.

Kim, B., R. J. Graves, S. S. Heragu and A. S. Onge (2002). "Intelligent agent based modeling of an industrial warehouse problem." IIE Transactions **34**(7): 601-612.

Lau, H. Y. K., V. W. K. Wong and I. S. K. Lee (2003). Immunity-based autonomous guided vehicles control, IOS Press.

Lin, Y. J. J. S. (1992). "Integrated Shop Floor Control Using Autonomous Agents." IIE Transactions **24**(3): 57-71.

Liu, S., W. A. Gruver and D. B. Kotak (2002). "Holonic coordination and control of an automated guided vehicle system." Integrated Computer-Aided Engineering **9**(3): 235-250.

Luck, M., P. McBurney and C. Preist (2003). Agent Technology - Enabling Next Generation Computing: A Roadmap for Agent Based Computing.

Maione, B. and D. Naso (2001). "Evolutionary adaptation of dispatching agents in heterarchical manufacturing systems." International Journal of Production Research **39**(7): 1481-1503.

Maturana, F., W. Shen and D. H. Norrie (1999). "Metamorph: An adaptive agent-based architecture for intelligent manufacturing." International Journal of Production Research **37**(10): 2159-2173.

McDonnell, P., G. Smith, S. Joshi and S. R. T. Kumara (1999). "A Cascading Auction Protocol as a Framework for Integrating Process Planning and Heterarchical Shop Floor Control." International Journal of Flexible Manufacturing Systems **11**: 37-62.

McElroy, J. F., L. M. Stephens, R. D. Bonnell and J. Gorman (1989). Communication and cooperation in a distributed automatic guided vehicle system. Proceedings of the IEEE Southeastcon '89.

Mes, M. R. K., M. C. van der Heijden and A. van Harten (2007). "Comparison of agent-based scheduling to look-ahead heuristics for real-time transportation problems." European Journal of Operational Research **181**(1): 59-75.

Padgham, L. and M. Winikoff (2004). Developing Intelligent Agent Systems: A Practical Guide, John Wiley and Sons Ltd.

Parunak, H. v. D. (1987). Manufacturing experience with the Contract Net. Distributed artificial intelligence. M. N. Huhns. London, UK, Pitman**:** 285-310.

Parunak, H. v. D. (1998). Industrial and Practical Applications of Agent-based Systems. Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence. G. Weiss. Cambridge, MIT Press**:** 377-424.

Parunak, H. V. D., A.D. Baker & S.J. Clark (2001). "The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design." Integrated Computer-Aided Engineering **8**(1): 45-58.

Shen, W. and D. H. Norrie (1999). "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey." Knowledge and Information Systems, an International Journal **1**(2): 129-156.

Shen, W. and D. H. Norrie (2001). "Dynamic manufacturing scheduling using both functional and resource related agents." Integrated Computer-Aided Engineering **8**(1): 17-30.

Silver, E. A., D.F. Pyke, R. Peterson (1998). Inventory management and production planning and scheduling. New York, Wiley.

Smith, R. (1980). "The Contract Net Protocol: A High Level Negotiation Protocol for Distributed Problem Solving." IEEE Transactions on Computers **29**.

Vickrey, W. (1961). "Counterspeculation, auctions, and competitive sealed tenders." Journal of Finance **16**(1): 8-37.

Wallace, A. (2001). "Application of AI to AGV control - agent control of AGVs." International Journal of Production Research **39**(4): 709-726.

Wood, M. F. and S. A. DeLoach (2000). An Overview of the Multiagent Systems Engineering Methodology. The First International Workshop on Agent-Oriented Software Engineering (AOSE-2000).

Wooldridge, M. (2002). MultiAgent Systems, Jonn Wiley & Sons, Ltd.

Wooldridge, M. and N. R. Jennings (1995). Agent Theories, Architectures, and Languages: A Survey. Proceedings of the ECAI-94 Workshop on Agent Theories, Architectures, and Languages. Amsterdam, The Netherlands**:** 1-39.

Wooldridge, M., N. R. Jennings and D. Kinny (2000). "The Gaia Methodology For Agent-Oriented Analysis And Design." Autonomous Agents and Multi-Agent Systems **3**(3): 285-312.

Appendix

Formally, we define an AGV schedule Ψ by an ordered list of 3-tuples (i,spt_i,sdt_i) where i refers to a specific job, spt_i to the scheduled pickup time and sdt_i to the scheduled delivery time of this job. We write $\tau(dest_i,ori_{i+1})$ for an empty move from the destination of job i to the origin of the next job. The value of a schedule is given by the deviations from the best rising times plus α times the tardiness and β times the total time:

$$V^a(\Psi) = \beta\left(sdt_{|\Psi|} - \theta\right) + \sum_{i=1}^{|\Psi|} \begin{cases} \left|sdt_i - b_i^p\right| + \alpha \cdot \max\left(0, sdt_i - l_i^p\right) & \textit{preparation job} \\ \left|sdt_i - b_i^d\right| + \alpha \cdot \max\left(0, sdt_i - l_i\right) & \textit{delivery job} \end{cases}$$

We introduce the symbol θ to indicate the current time. The scheduled delivery time is given by $sdt_i = spt_i + h(i)$, where *h(i)* is the handling time of job *i*. In case of a preparation job, this handling time is given by the expected time between picking up a barrel at the storage area and dropping the barrel at the rising area, including expected waiting times at the silos and the mixer. In case of a delivery job, the handling time is given by the time between picking up a barrel at the rising area dropping it at the line. The pickup times are scheduled such that they minimize the total costs of a schedule:

$$\min_{spt_i, i=2..|\Psi|} V^a(\Psi)$$

$$s.t.$$

$$spt_i \geq e_i \quad for\ all\ delivery\ jobs$$

$$spt_i \geq spt_{i-1} + h(i-1) + \tau(dest_{i-1}, ori_i) \quad for\ i \geq 2$$

Here we assumed that the scheduled times of the first job (which may be in execution) may not be changed.